11

Supporting Interactions with Multiple Platforms Through User and Task Models

Luisa Marucci, Fabio Paternò, and Carmen Santoro

ISTI-CNR, Italy

11.1. INTRODUCTION

In recent years, interest in model-based approaches has been increasing. The basic idea of such approaches is to identify useful abstractions highlighting the main aspects that should be considered when designing effective interactive applications. UML [Booch *et al*. 1999], the most common model-based approach in software engineering, has paid very little attention to supporting the design of the interactive component of software. Therefore, specific approaches have been developed for interactive system design. Of the relevant models, task models play a particularly important role because they indicate the logical activities that an application should support. A task is an activity that should be performed in order to reach a goal. A goal is either a desired modification of state or an inquiry about the current state.

For the generation of multiple user interfaces, task models play a key role in the adaptation to different contexts and platforms. The basic idea is to capture all the relevant

Multiple User Interfaces. Edited by A. Seffah and H. Javahery
 ISBN: 0-470-85444-8

requirements at the task level and then use this information to generate effective user interfaces tailored for different types of platforms. Information about the design of the final user interface can be derived from analysing task models. For example, the logical decomposition of a task can provide guidance for the generation of the corresponding concrete user interface. The task structure is reflected in the graphical presentation by grouping together interaction techniques and objects associated with the same sub-task. We have identified a number of possibilities for how tasks should be considered in the generation of multi-platform user interfaces, for example:

- When the same task can be performed on multiple platforms in the same manner;
- When the same task is performed on multiple platforms but with different user interface objects or domain objects;
- When tasks are meaningful only on specific platforms.

In addition to adapting interfaces at the design phase, it is possible to adapt them at run-time by considering users' preferences and environment (location, surrounding, etc.). Preference and environment information is used to adapt the navigation, presentation and content of the user interface to different interaction platforms (see Figure 11.1).

Figure 11.1. User model support for multiple platforms.

User modelling [Brusilovsky 1996] supports adaptive interfaces that change according to user interaction. It can also be helpful in designing for multiple interaction platforms. User models represent aspects such as knowledge level, preferences, background, goals, physical position, etc. This information provides user interfaces with adaptability, which is the ability to dynamically change their presentation, content and navigation in order to better support users' navigation and learning according to the current context of use. Several types of user models have been used. For example, some models use information about the level of users' knowledge for the current goals; other models employ user stereotypes and evaluate the probability of their relevance to the current user.

Various aspects of user interfaces can be adapted through user models. Text presentation can be adapted through techniques such as conditional text or stretch-text (text that can be collapsed into a short heading or expanded when selected). User navigation can also be adapted using techniques such as direct guidance and adaptive ordering and hiding of links. Adaptive techniques have been applied in many domains. Marucci and Paternò [2002] describe a Web system supporting an adaptive museum guide that provides virtual visitors with different types of information related to the domain objects presented (introduction, summary, comparison, difference, curiosity) according to their profile, knowledge level, preferences and history of interactions.

To date, only a few publications have considered user modelling to support the design of multi-platform applications, and there are still many issues that need to be solved in this context. For example, Hippie [Oppermann and Specht 2000] describes a prototype that applies user modelling techniques to aid users in accessing museum information through either a web site or a PDA while they are in the museum. In our approach we also consider the use of mobile outdoor technologies and provide user models integrated with task models developed in the design phase.

In this chapter we first introduce a detailed scenario to illustrate the type of support that we have designed. Then we describe our method, how the user model is structured and how such information is used to obtain an adaptive user interface. We also discuss the types of rules for using information in the user model to drive adaptive behaviour. Finally, we provide some concluding remarks.

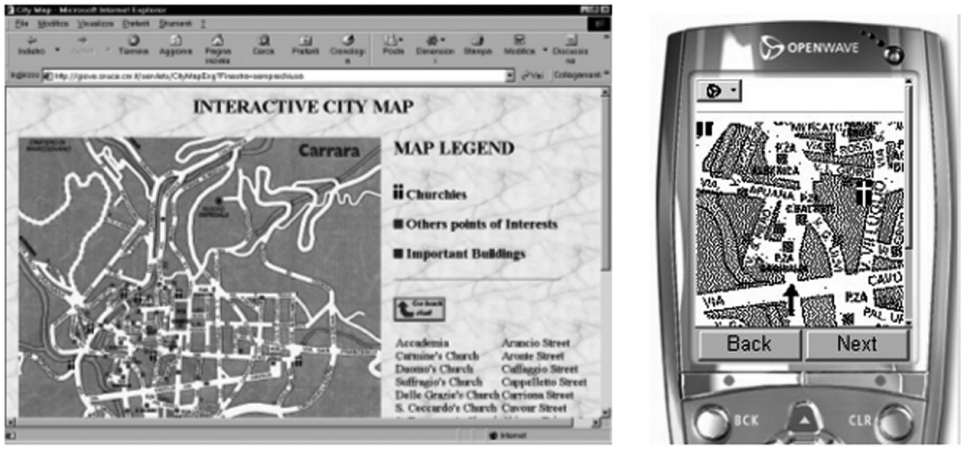
11.2. AN ILLUSTRATIVE SCENARIO

In this section we provide the reader with a concrete example of the type of support that our approach provides. The scenario describes an application that provides an interactive guide to a town.

From a desktop computer at the hotel, John visits the Carrara web site. He finds it interesting. In particular, he is interested in marble sculptures located close to Piazza Garibaldi. He spends most of his time during the virtual visit (Figure 11.2a) accessing the related pages and asking for all the available details on such artworks.

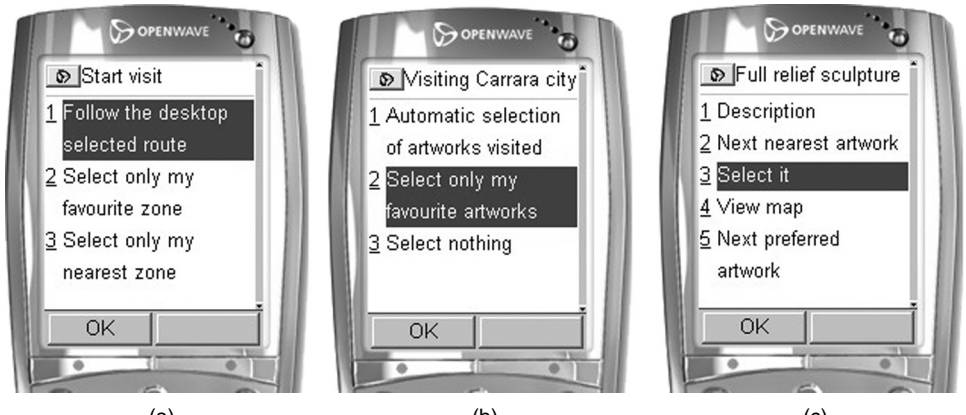
The next day, John leaves the hotel and goes to visit the historic town center. When he arrives he accesses the town's web site through his phone.

The system inherits his preferences and levels of knowledge from the virtual visits performed in the hotel. Thus, it allows him to access information on the part of the town



(a) (b)

Figure 11.2. Spatial information provided through (a) the desktop and (b) the cell phone.



(a) (b) (c)

Figure 11.3. Cell phone support during the visit to the historic town.

that interests him most (Figure 11.3a) and navigation is supported through adaptive lists. These lists are based on a ranking determined by the interests expressed in the previous visit using the desktop system. During the physical visit he sees many works of art that impress him, but there is no information available nearby, so he annotates them through the phone interface (Figures 11.3b and 11.3c).

In the evening, when he is back in the hotel, John accesses the town web site again through his login. The application allows him to access an automatically generated guided tour of the town (Figure 11.4) with an itinerary based on the locations of the works of art

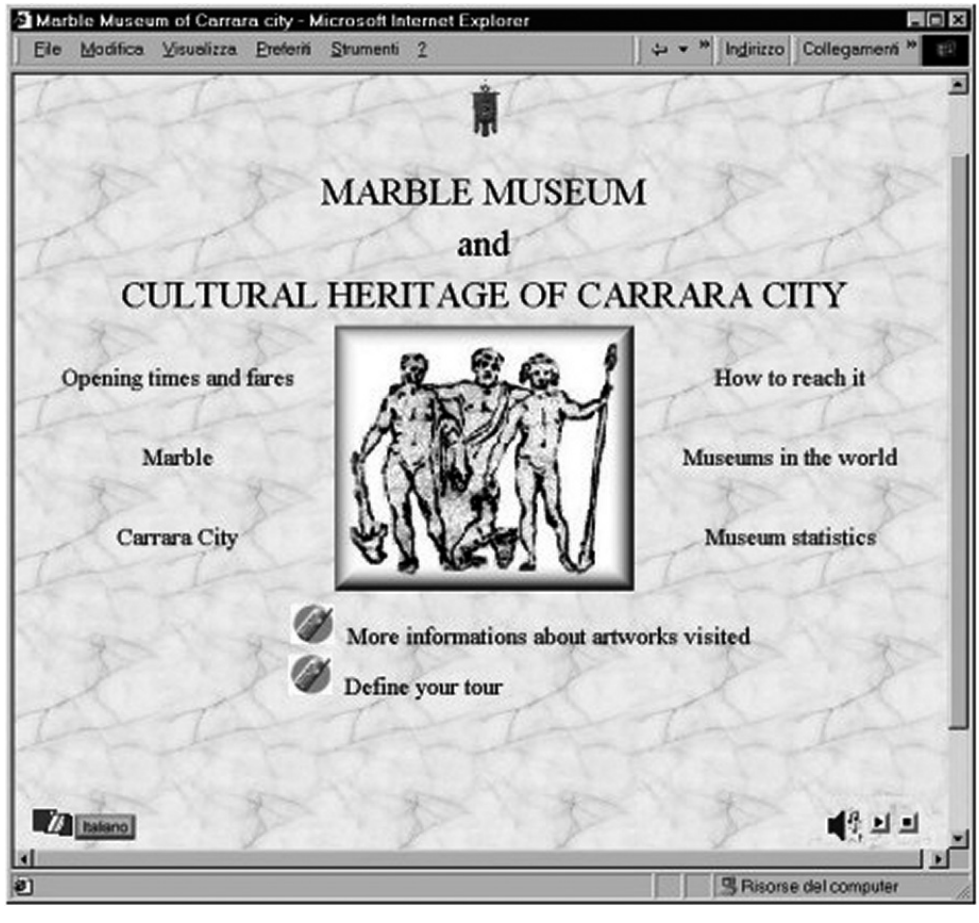


Figure 11.4. User interface to the desktop system after access through phone.

that impressed him. He can modify the tour if he no longer finds some of the proposed works of art interesting. In this way, he can perform a new visit of the most interesting works of art, receiving detailed information about them.

11.3. GENERAL DESCRIPTION OF THE APPROACH

In order to support the development of systems that adapt to the current user, device and context, we use the models shown in Figure 11.5. In this diagram we consider both the static development of interactive systems for a set of platforms and the dynamic adaptation of interactive systems to changes in context in real time.

- In the static case, a system specification in terms of the supported tasks is used to create an abstract version of a user interface (including both abstract presentation and dialogue

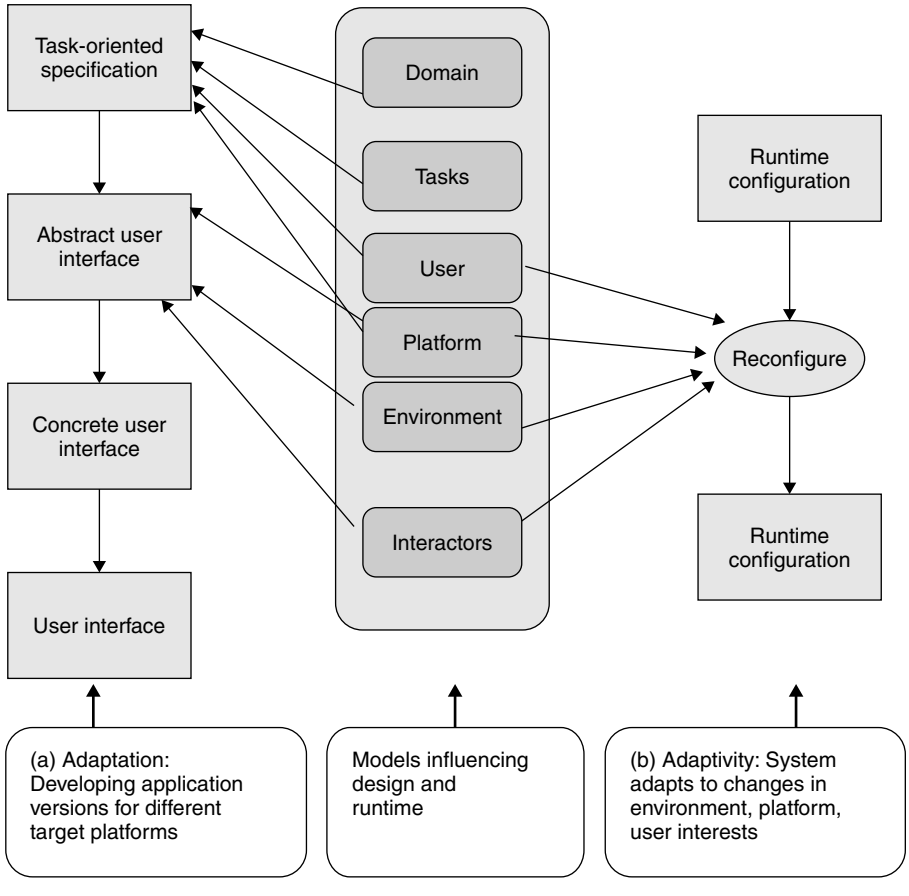


Figure 11.5. Models considered at design time and run-time.

design); from the abstract user interface, a concrete user interface is derived, in which the abstract interaction mechanisms are bound to platform-specific mechanisms.

- In the dynamic case, external triggers lead to the real-time reconfiguration of the interactive system during use. These triggers can be user actions (e.g., connecting a PDA to a mobile phone to provide a network connection) or events in the environment (e.g., changing noise and light level as a train enters a tunnel, or network failure).

In order to better describe the static development and the dynamic reconfiguration of systems, we refer to a number of models:

- The *Task Model* describes a set of activities that users intend to perform while interacting with the system. We can distinguish two types of task models: the system task model, which is how the designed system requires tasks to be performed, and the user

task model, which is how users expect to perform their activities. A mismatch between these two models can generate usability problems.

- The *Domain Model* defines the objects that a user can access and manipulate in the user interface. This model also represents the object attributes and relationships according to semantically rich expressions.
- *Interactors* [Paternò and Leonardi 1994] describe the different interaction mechanisms independent of platform (e.g., the basic task an interactor is able to support). The interactor model operates primarily at the level of the abstract description of the user interface.
- The Platform Model describes the physical characteristics of the target platforms, for example characteristics of the available interactive devices such as pen, screen, voice input and video cameras.
- The *Environment Model* specifies the user's physical environment.
- The *User Model* describes information such the user's knowledge, interests, movements and personal preferences.

We have identified a set of design criteria for using logical information in the models to generate multimedia user interfaces adapted to a specific user, platform and context of use. For a given task and device, these design criteria indicate, for example, which interaction and presentation techniques are the most effective in a specific configuration setting, and how the user interface should adapt to a change of device or environmental conditions.

In the following sections we will show how these models take part in the overall design process. The discussion will focus on the task model and user model, describing the role they play in the generation of user interfaces that adapt to changes in context. Afterward, we will describe their relationships in more detail, and we will present an example that helps to explain the approach.

11.4. ROLE OF THE TASK MODEL IN DESIGN

The design of multi-platform applications can employ different approaches. It is possible to support the same type of tasks with different devices. In this case, what has to be changed is the set of interaction and presentation techniques to support information access while taking into account the resources available in the device considered. However, in some cases designers should consider different devices also with regard to the choice of tasks to support. For example, phones are more likely to be used for quick access to limited information, whereas desktop systems better support browsing through large amounts of information. Since the different devices can be divided in clusters sharing a number of properties, the vast majority of approaches considers classes of devices rather than single devices. On the one hand, this approach tends to limit the effort of considering all the different devices. On the other hand, different device types might be needed because of the heterogeneity of devices belonging to the same platform.

The fact that devices and tasks are so closely interwoven in the design of multiplatform interactive applications is a central concern running through our method, which is composed of a number of steps allowing designers to start with an overall envisioned

Figure 11.6. Deriving multiple user interfaces from a single task model.

task model of a nomadic application and then derive effective user interfaces for multiple devices (see Figure 11.6). The approach involves four main steps:

1. High-level task modelling of a multi-context application: In this phase, designers define the logical activities to be supported and the relationships among them. They develop a single model that addresses the various contexts of use and roles; they also develop a domain model to identify the objects manipulated in tasks and the relationships among such objects. Such models are specified using the ConcurTaskTrees (CTT) notation. The CTT Environment tool [Mori *et al*. 2002] publicly available at http://giove.cnuce.cnr.it/ctte.html supports editing and analysis of task models using this notation. The tool allows designers to explicitly indicate the platforms suitable to support each task.
2. Developing the system task model for the different platforms: Here designers filter the task model according to the target platform and, if necessary, further refine the task model for specific devices. In this filter-and-refine process, tasks that cannot be supported on a given platform are removed and the navigational tasks necessary to interact with the platform are added. In other cases it is necessary to add supplementary details on how a task is decomposed for a specific platform.
3. From system task model to abstract user interface: Here the goal is to obtain an abstract description of the user interface. This description is composed of a set of abstract presentations that are identified through an analysis of the task relationships. These abstract presentations are then structured by means of interactors (see Section 3 for definition). Then we identify the possible transitions among the user interface

presentations as a function of the temporal relationships in the task model. Analysing task relationships can be useful for structuring the presentation. For example, the hierarchical structure of the task model helps to identify interaction techniques and objects to be grouped together, as techniques and objects that have the same parent task are logically more related to each other. Likewise, concurrent tasks that exchange information can be better supported by highly integrated interaction techniques.

4. User interface generation: This phase is platform-dependent and device-dependent. For example, if the platform is a cellular phone, we also need to know the type of micro-browser supported and the number and types of soft-keys available in the specific device considered.

In the following sections we discuss these steps in detail. We have defined XML versions of the language for task modelling (ConcurTaskTrees) and the language for modelling abstract interfaces; we have also developed automatic transformations among these representations.

11.4.1. FROM THE TASK MODEL TO THE ABSTRACT USER INTERFACE

The task model is the starting point for defining an abstract description of the user interface. This abstract description has two components: a presentation component (the static structure of the user interface) and a dialogue component (the dynamic behaviour).

The shift from task to abstract interaction objects is performed through three steps:

1. Calculation of Enabled Task Sets (ETS): the ETSs are sets of tasks enabled over the same period of time according to the constraints indicated in the task model. They are automatically calculated through an algorithm that takes as input (i) the formal semantics of the temporal operators of the CTT notation and (ii) a task model. For example, if two tasks t1 and t2 are supposed to be concurrently performed, then they belong to the same ETS; they can be performed in any order so their execution will be enabled over the same period of time. If they are supposed to be carried out following a sequential order (first t1 then t2), they cannot belong to the same ETS since the performance of t2 will be enabled only after the execution of t1; thus they will never be enabled during the same interval of time. The need for calculating ETSs is justified by the fact that the interaction techniques supporting the tasks belonging to the same enabled task set are logically candidates to be part of the same presentation. In this sense, the ETS calculation provides a first set of potential presentations. Furthermore, the calculation of the ETS implies the calculation of the conditions that allow passing from ETS to ETS–we called them ‘transitions’.
2. Heuristics for optimizing presentation sets and transitions: these heuristics help designers reduce the number of presentations considered in the final user interface. This is accomplished by grouping together tasks belonging to different ETSs. In fact, depending on the task model the number of ETSs can be rather high. As a rule of thumb, the number of ETSs is of the same order as the number of enabling operators in the task model. So, in this phase we specify rules (heuristics) to reduce their number by merging two or more ETSs into new sets, called Presentation Task Sets (PTS).

3. Mapping presentation task sets and their transitions onto sets of abstract interaction objects and dialogue: a number of rules have been identified in order to perform the mapping between a task and a suitable abstract user interface object. These rules are based on the analysis of the multi-dimensional information associated with tasks – for example, the goal, the objects manipulated and the frequency of the task. Each dimension functions as a sort of condition during the visit of the tree-like structure of the language describing interactors (see Figure 11.7), in order to select the most suitable one. The transitions between different presentation sets are directly mapped into connections linking the different presentations of the abstract user interface.

All of these transformations are supported by our TERESA tool [Mori *et al*. 2003] publicly available at http://giove.cnuce.cnr.it/teresa.html.

11.4.2. THE LANGUAGE FOR ABSTRACT USER INTERFACES

The set of presentation sets obtained in the previous step is the initial input for building the abstract user interface specification. This specification is composed of interactors or Abstract Interaction Objects (AIOs) associated with the basic tasks. Such interactors are high-level interaction objects that are classified first by type of basic task supported, then by type and cardinality of the associated objects and lastly by presentation aspect.

Figure 11.7 shows that an interface is composed of one or more presentations and each presentation is characterised by an aio or an aio_composition and 0 or more connections. There are two main types of objects in the abstract user interface: elementary abstract interaction objects (aio) and complex expressions (aio_composition) derived from applying the operators to these interaction objects. While the operators describe the static organisation of the user interface (in the next section we provide more detail on them),

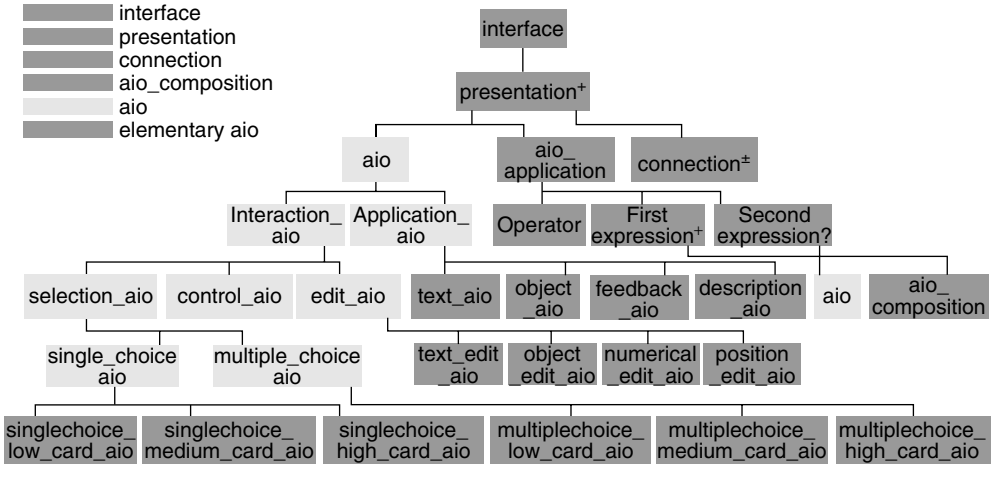


Figure 11.7. The tree-like representation of the language for specifying the abstract user interface.

the set of connections describes how the user interface evolves over time, namely its dynamic behaviour.

11.4.2.1. From Presentation Task Sets to Abstract User Interface Presentations

The abstract user interface is mainly defined by a set of interactors and the associated composition operators. The type of task supported, the type of objects manipulated and their cardinality are useful elements for identifying the interactors. In order to compose such interactors we have identified a number of composition operators for designing usable interfaces. These composition operators are associated with communication goals that designers aim to achieve [Mullet and Sano 1995]:

- *Grouping (G)*: The objective is to group together two or more elements, so this operator should be applied when the involved tasks share some characteristics. A typical situation is when the tasks have the same parent task. This is the only operator for which the position of the different operands is irrelevant.
- *Ordering (O)*: This operator is applied when some kind of sequential order exists among elements. The most typical sequential order is the temporal order. The order in which the different elements appear within this operator reflects the order within the group.
- *Relation (R)*: This operator is applied when a relation exists between n elements y_i, $i = 1, \ldots, n$ and one element x. In the task model, a typical situation is when a leaf task t is at the right-hand side of a disabling operator. In this case all the tasks that could be disabled by t (at whatever task tree level) are in relation to t. This operator is not commutative.
- *Hierarchy (H)*: This operator means that a hierarchy exists among the involved interactors. The importance level associated with the operands identifies the degree of visual prominence that the associated interaction objects should have in the user interface. The degree of importance can be derived from the frequency of access or from details of the application domain. Various techniques can be used to convey importance. In graphical user interfaces, one method is allotting more screen space to objects that are hierarchically more important.

These operators are applied to tasks belonging to the same PTS, depending on the temporal relationships among those tasks. The temporal relationships are derived from the task model in the following manner: if the two concerned tasks are siblings, the temporal relationship is represented by the CTT operator existing between them; if this is not the case (e.g. the two tasks have different parent tasks) the temporal relationship is easily derived because temporal relationships between tasks are inherited by their subtasks.

11.4.2.2. The Dialogue Component

In specifying the dynamic behaviour of the abstract user interface, an important role is played by abstract interaction objects associated with the transitions. For each presentation task set P, *transition(P)* specifies the conditions allowing for the transition of the abstract user interface from the current presentation task set P into another presentation task set P'. The transitions can directly correspond to tasks, or, alternatively, can be expressed by

means of a Boolean expression. For example, when we want to express that more than one task has to be executed in order to trigger the activation of a different presentation, an AND operator combines the tasks.

11.4.3. FROM THE ABSTRACT USER INTERFACE TO ITS IMPLEMENTATION

Once the elements of the abstract user interface have been identified, each interactor is mapped onto interaction techniques supported by the specific device configuration (operating system, toolkit, etc.). For example if the object of the abstract user interface allows for a single selection from a set of objects, various implementations are available to the designer depending on the capabilities of the platform or device in question; these can include radio button menus, pull-down menus, list menus, etc.

In addition, since relationships between interactors are expressed with composition operators, they have to be appropriately implemented in order to convey their logical meaning in the final user interface. Several techniques are available for this purpose. For instance, in graphical user interfaces, a typical example is the set of techniques for conveying groupings by using classical presentation patterns such as proximity, similarity and continuity. If a different modality is used, the meaning of the same operators should be conveyed through different mechanisms. For example, in audio user interfaces, we would convey groupings with aural attributes such as pitch and volume.

As another example, a hierarchy operator for textual objects in a graphical user interface could represent important objects with larger fonts, whereas in an audio-based user interface, the hierarchy operator could represent important verbal information with a higher volume.

11.5. RELATIONS BETWEEN TASK MODEL AND USER MODEL

In our approach we assume that a model-based method has been followed in the design of the multi-platform application. As noted earlier in this chapter, the ConcurTaskTrees notation [Paternò 1999] allows designers to develop task models of nomadic applications. This means that in the same model, designers can describe tasks to be performed on different platforms and their interrelationships [Paternò and Santoro 2002]. From this high level description it is possible to obtain first the system task model associated with each platform and then the corresponding device-level user interface. The task model can also be expressed in XML format.

In our case we use the XML specification as input for the creation of the user model that will be used for adaptivity at run-time. The two models share some information, but also contain different elements. This means that some elements of the task model are removed and others added in order to make the two models compatible. In addition, the user model is mainly characterised by values that are updated dynamically based on users' interactions with the interface. For each user, the user model is updated when the user interacts with any of the available platforms. A run-time support algorithm uses the

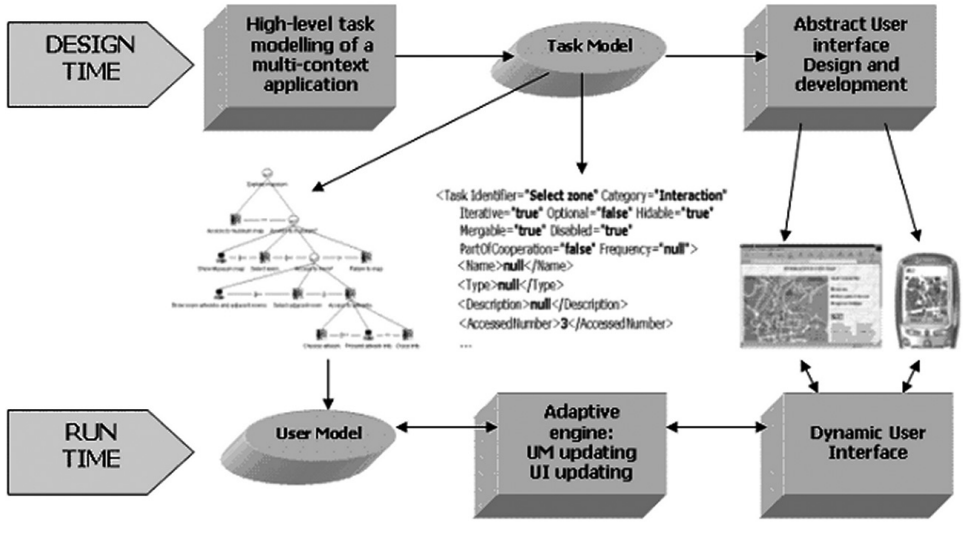


Figure 11.8. Relationships between task model and user model.

user model to modify the user interface presentation, navigation and content by applying previously defined adaptivity rules.

One advantage of this approach is that the task model developed at design time already provides useful information for run-time adaptive support (Figure 11.8). This information from the task model at design time includes:

- The temporal dependencies among tasks performed on different platforms;
- The tasks that can be performed through multiple platforms;
- The association of tasks with domain objects and related attributes;
- The definition of objects and attributes accessible through a given platform.

The performance of some tasks (from either phone or desktop) can change the level of interest associated with some domain objects (for example the preferred city zone), and this information can also be used to adapt the presentation support for a platform different from that currently in use (for example, the order of the links in a list).

11.6. THE USER MODEL

In our approach, the user model is structured in such a way as to indicate user preferences and acquired knowledge depending on the user's access to the application. Referring to the scenario of use of the Carrara Web site in section 2:

- User preferences can include, for example, the preferred city zone, navigation style, theme or features of an artwork.

- Acquired knowledge can include, for example, the level of knowledge about an author, a historical period or a material.
- The general format of the user model (in XML file format) includes:

As we can see in Figure 11.9, the user model is tightly related to the task model. It contains information that is dynamically updated such as the number of times that a task has been performed or that an object has been accessed. It also contains fields that allow dynamic modification of the availability of performing a specific task: *Mergeable* indicates whether to merge the execution of a task with a different task, *Hideable* indicates whether to hide its performance in another, more general task, and *Disabled* indicates whether to completely disable it for the current user.

For each task, all the attributes listed above can be defined (through a dedicated tool), including the properties related to adaptive support (Figure 11.10). After this step, the tool generates the XML file in the following general form (Figure 11.11):

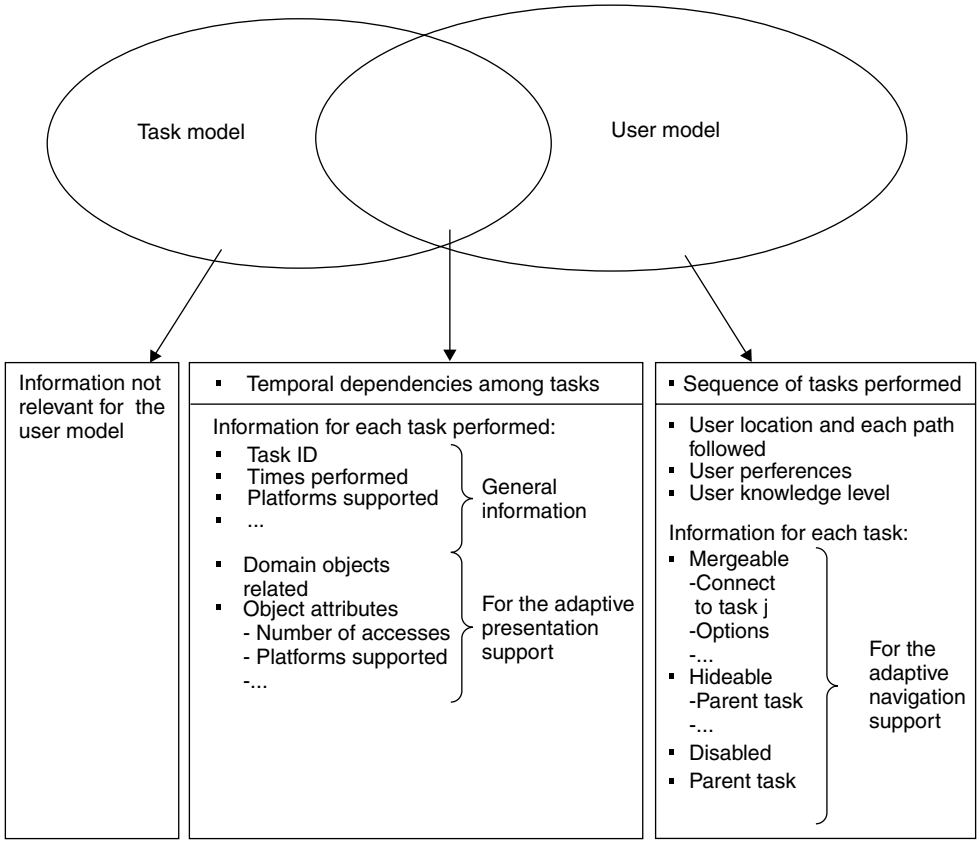


Figure 11.9. Information contained in the user model and its relation to the task model.

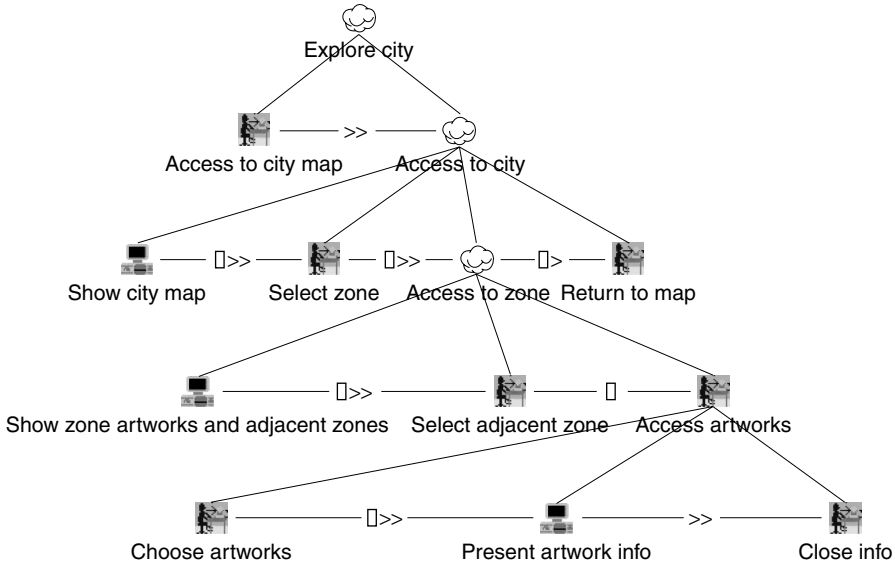


Figure 11.10. Example of a task model.

```
<Task Identifier="Select zone" Category="Interaction"
    Iterative="true" Optional="false" Hidable="="true"
    Mergable="true" Disabled="true"
    PartOfCooperation="false" Frequency="null">
    <Name>null</Name>
    <Type>null</Type>
    <Description>null</Description>
    <AccessedNumber>3</AccessedNumber>
    ...
```

Figure 11.11. An excerpt of the XML file containing information about task models.

This file is updated during the user session. From analysis of the file, the system is able to determine the tasks performed by the user and their sequence, as well as the object classes and related subclasses. From this user input, the system computes the navigation preferences by analysing information such as the sequence of tasks, the tasks never performed, and the tasks most frequently performed. The system also evaluates the presentation preferences by analysing the objects' classes and subclasses.

The location is an attribute related only to mobile interactive platforms. For example if the user has accessed the system by mobile phone, then after the user selects an item from the *Materials* list, the system offers the option of displaying only the artworks made of local materials.

The domain model is structured in terms of object classes and related subclasses that are manipulated during task performance. The relationships between tasks and domain objects are represented in the user model. The association between tasks and object instances can

be either static or dynamic. For example, in the task of selecting an element from a list of predefined values, the association is static, whereas in the task of displaying information on a work of art whose name is provided by the user, the association is dynamic.

The domain objects that can be accessed and manipulated vary by device. In general, domain objects that can be manipulated by phone are more limited than those accessible via desktop computers and have different spatial attributes related to the user position, such as *closeness*.

Likewise, the supported tasks depend on the interaction platform. For example, some tasks are associated with a virtual visit on a desktop computer and others are associated with access by mobile phone. In addition, performance of certain tasks on one platform may depend on the accomplishment of other tasks through other devices (for example the desktop task of reviewing an itinerary previously annotated with a phone device).

In response to the user's behaviour in real time, the user model dynamically updates user knowledge and preferences. This has the effect of updating objects, attributes and task performance frequencies. The application can dynamically change the supported navigation according to the frequency of performance of certain tasks and the frequency of use of certain objects.

11.7. ADAPTIVE RULES

This section describes the rules that are used to drive the adaptivity of the user interface. In the next subsection we will explain how these rules are handled, and how they result in adaptive navigation and presentation as a function of the users' interactions with the system on different platforms. In particular we will examine examples of the adaptation of navigation, presentation and content of the user interface. The following tables (Tables 11.1, 11.2 and 11.3) show when a rule comes into force and the effect on interactive system behaviour. It is possible to relate such rules to the identification of interaction patterns directly from the end-user experience [Seffah and Javahery 2003].

11.7.1. NAVIGATION AS A FUNCTION OF TASK FREQUENCY

Here we discuss how the system handles the situations where the user always repeats the same sequence of tasks. For example, we can consider when the user selects a set of domain objects associated with a general topic and then a more refined subset iteratively (see Figure 11.12).

The recurrent selection of a specific type of artwork (e.g. made of bronze, defined as full relief sculpture, etc.), followed by a more specific selection (e.g. bronze artworks from the 20th century, full relief sculpture by the artist Vatteroni, etc.) causes the appearance in the interface of a new link for direct access to the subclass: 'Bronze artworks in XX Century' or 'Vatteroni's full relief sculpture'. This link will appear until the user has visited all the artworks belonging to that subset or until the system detects different preferences.

We can follow the corresponding changes in the user model: for each task there is an attribute that represents the possibility of that task's being *merged*, an indication of the

Table 11.1. Rules for adaptive navigation.

If...	Then...
The user always performs the same sequence of tasks in order to reach a goal	Change the navigation support so as to reduce the time required to achieve the goal
The user performs a task on one platform and then accesses the application through another platform	Change the user model state to enable or disable certain tasks
The user never selects a task (for example, a link selection) during one or more sessions on any platform	Hide the task support from all platforms (for example, remove link)
Mobile context: The user is near an object of interest in the physical world	Advise users through their mobile device
Mobile context: The user is following a physical path in the environment	Determine the next object of interest for users based on their preference and location
The user often selects a domain object set that satisfies a given rule (for example belonging to a city zone, with the same characteristics, etc.)	Change navigation modality so as to enable the related tasks.
The user never selects a domain object set that satisfies a given rule (for example belonging to a city zone, with the same characteristics, etc.)	Change navigation modality so as to disable the related tasks.

Table 11.2. Rules for adaptive presentation.

If...	Then...
The user often selects a domain object (independently of the task order and platform)	Provide access to this object or attribute in a high priority position.
The user never selects a domain object (independently of the task order and platform)	Provide access to this object or attribute in a low priority position.
The user often performs the same type of tasks	Change the presentation according to the most frequently used task types

task to which it can be connected and the new name to be given to this unified task as well as the number of accesses, object instance and object subsets selected.

In the previous example (the recurrent selection of bronze artworks and then bronze artworks from the XX Century), this will generate a link *Bronze artworks in XX Cen.*, in both the desktop and phone interfaces in which the user can select the material. During dynamic generation of the user interface, the system first analyses the XML file content and then generates the links.

Table 11.3. Rules for adaptive content.

If...	Then...
The user has already seen a specific domain object and then accesses a similar object	Change the content so as to explain the difference or similarity as compared to the previously seen object
The user shows advanced knowledge of a certain topic	Increase the detail in the description of the elements of interest, within the constraints of the current device

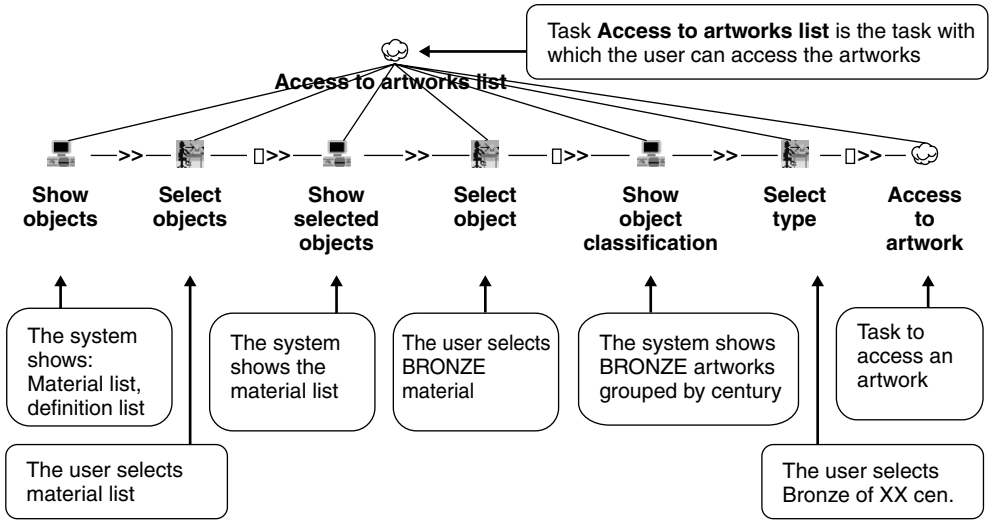


Figure 11.12. A task model for two-stage selection of objects.

Another example is when the user never performs certain tasks during a session or during different sessions. In this case the system will remove the tasks in question. Thus, if the task is never performed over one or more sessions in any platform (assuming that it is defined for multiple platforms), it can be disabled by setting the corresponding attribute.

11.7.2. NAVIGATION AS A FUNCTION OF TASK PERFORMANCE

Tasks performed in a specific platform can generate a change in the task model for another platform. For example, let us consider a scenario where the user previously selects a tour on a desktop computer, indicates preferences for a city zone and then accesses the application through a cell phone.

When the user selects a tour on the desktop computer, via either a map or the predefined link, the task *Follow the desktop selected route* in the user model will be modified (see Figure 11.13). The corresponding *Disabled* attribute, previously set to true, will be set to false, and the corresponding object instance will be the tour chosen by the user.

(a) (b) (c)

Figure 11.13. Access to the application (a) for the first time, (b) after a desktop visit that selected a tour and (c) after a desktop visit that did not select a tour.

For the reverse case, from the mobile platform the user chooses the option of selecting the artworks seen during the visit, so as to view descriptions and details later on at home using the desktop computer. This will enable the task *More Information about artworks visited* in the desktop platform, and each of the artworks selected will be added as the objects corresponding to that task.

11.7.3. MODIFICATION OF PRESENTATION

The following example demonstrates a change in presentation for a task whose objects are the artworks located in the historic city center. The user can access these artworks by choosing one of the following alternatives: Streets, Buildings, Churches, or Squares. Suppose that the user often chooses 'Streets'. The user model contains the choice task whose objects correspond to the artworks of the city, along with the specific platforms from which each object can be accessed.

More generally, the user model also contains the objects manipulated by each task as well as the platforms supporting each object. For each user choice, the system stores the objects selected in the user model. In the example mentioned above, the user first selects 'artworks in Carrara city' and then the object 'Streets'. The recurrent choice of this attribute will cause a change in the order of items in the corresponding list (see Figure 11.14).

In summary, if the user selects an object on one platform, this will cause a change in the sequence of all lists containing that object, across all platforms.

11.7.4. MODIFICATION OF CONTENT PRESENTATION

In one of the rules in Table 11.1, if the user frequently accesses a domain object, this causes a modification of the content presentation and an updating of the user's knowledge

Figure 11.14. An adaptive list.

Figure 11.15. The user frequently asks for more information; the system generates it automatically.

level (while maintaining the same navigation path). For example, we can consider a scenario where the user accesses the description of an artwork and frequently asks for more information about that work.

The solution consists of introducing the ability to perform the same low-level task in the task hierarchy without performing the intermediate tasks. In the example in Figure 11.15, this means that the user can directly access detailed information about a work of art.

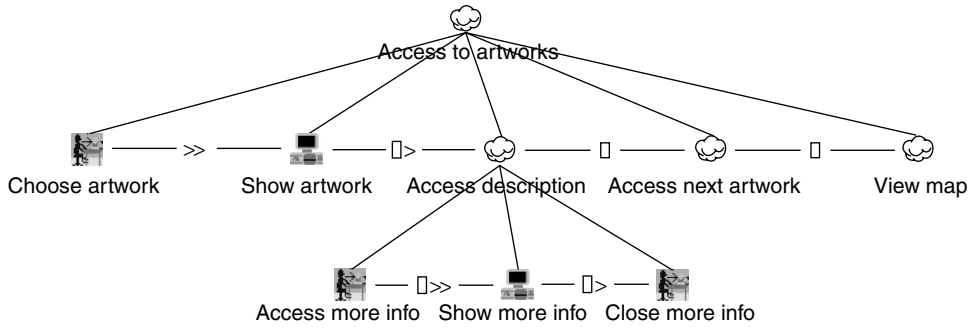


Figure 11.16. The task model for adaptive content.

Figure 11.16 shows the resulting task model. For each task that can be *Hideable* and is performed multiple times, we can conceal that task within the *Show Artwork* task. In the example, this means that when the system shows information on the artwork, it already includes detailed information. At the same time, the knowledge level of the user is updated (the user always accesses more information). When the user accesses the system from any platform, the knowledge level will be inherited.

In order to avoid user misunderstandings or confusion because of the adaptive support, it is possible to clearly indicate what part of the user interface is adaptive. For example, in a cell phone, a soft key can highlight the individual adaptive links or call up an adaptive list of frequently accessed links.

11.8. CONCLUSIONS

This chapter discussed how to provide adaptive support for multiple platforms based on task and user modelling techniques. The method was illustrated through a case study in the museum application domain.

In particular, this chapter addressed the use of task models at design time and their relationships with user models. A set of rules was introduced, based on the user model, for modifying presentation and dialogue as a function of users' interactions on different platforms. These rules allow applications to better support users' goals.

Future work will be dedicated to analysing in more detail whether adaptivity, especially in mobile phones, can sometimes disorient users. We will perform studies to determine how to introduce adaptivity in a way that avoids disorientation.

ACKNOWLEDGEMENTS

This work has been partially supported by the CAMELEON project (http://giove.cnuce.cnr.it/cameleon.html).

REFERENCES

Booch, G., Rumbaugh, J., and Jacobson, I. (1999) *Unified Modeling Language Reference Manual*. Addison Wesley.

Brusilovsky, P. (1996) Methods and techniques of adaptive hypermedia. *User Modelling and User Adapted Interaction*, 6(2–3), 87–129. URL: http://www.cntrib.andrew.cmu.edu/plb/UMUAI.ps.

Mori, G., Paternò, F., and Santoro, C. (2002) CTTE: Support for Developing and Analysing Task Models for Interactive System Design. *IEEE Transactions on Software Engineering*, 28 (8), 797–813.

Mori, G., Paternò, F., and Santoro, C. (2003) Tool Support for Designing Nomadic Applications, *Proceedings of ACM IUI 2003 International Conference on Intelligent User Interfaces*, January 12–15, 2003, Miami, FL, USA, 141–8. ACM Press.

Mullet, K., and Sano, D. (1995) *Designing Visual Interfaces*. Prentice Hall.

Oppermann, R., and Specht, M. (2000) A Context-Sensitive Nomadic Information System as an Exhibition Guide. *Proceedings of the Handheld and Ubiquitous Computing Second International Symposium*, Bristol, UK, September 25–27, 2000, 127–42.

Paternò, F. (1999) *Model-based Design and Evaluation of Interactive Applications*. Springer Verlag.

Paternò, F., and Leonardi, A. (1994) A Semantics-based Approach to the Design and Implementation of Interaction Objects. *Computer Graphics Forum*, 13(3), 195–204.

Paternò, F., and Santoro, C. (2002) One Model, Many Interfaces. *Proceedings of the 4th International Conference on Computer-Aided Design of User Interfaces CADUI 2002*, May 15–17, 2002, Valenciennes, Belgium, 143–54. Kluwer Academics, Dordrecht.

Seffah, A., and Javahery, H. (2003) Multiple User Interfaces: Definitions, Challenges and Research Opportunities, Chapter 2 in this book.